

DESIGN PROPOSAL : CONSTRUCTION OF A FREELINING FOUR-LEGGED QUADROPOD

Carolina de Freitas

Paige Pruitt

E155

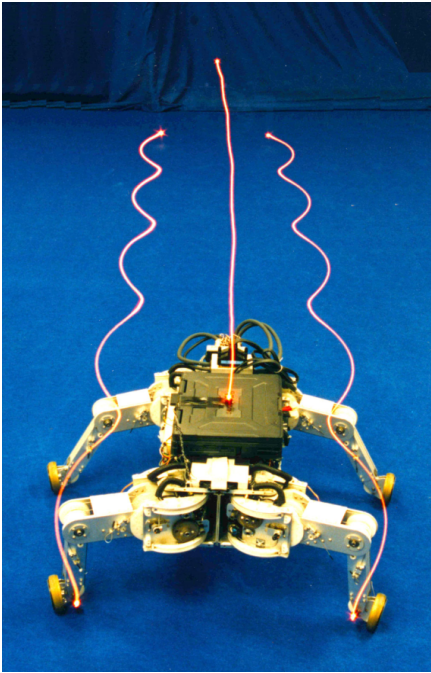


FIGURE 1 shows the roll-walk robot with LEDs attached to track the motion of the body and feet of the robot through space (HIROSE-FUKUSHIMA ROBOTICS LAB)

INTRODUCTION

This project involves the construction of a four-footed robot on wheels. Figure 1 shows a similar robot with its trajectory marked by LED lights over time (Leg-Wheel Hybrid Walking Vehicle "Roller-Walker"). The "Roller Walker" pictured here is the motivation for this project, it is a quadropod robot who "walks" in order to traverse treacherous terrain and can also turn its feet on its side to reveal four wheels which allow the robot to glide across a smooth floor. The rolling capability ensures that the robot can move as efficiently as possible regardless of the terrain it is covering.

For this project, the team will examine only the rolling motion of the hybrid "Roller-Walker" robot. This simplification adds a constraint (there is always a point on the wheel in contact with the ground) and makes the project more attainable.

The team is using several conference proceedings from the Hirose-Fukushima Lab to infer details about the robot. Many of the calculations and assumptions for the Roller-Walker hold true for the team's creation.

A second inspiration for this project comes from Free Line Skates™ (Farrelly). In particular, the movements of the Four Legged Quadropod's (FLQ) wheels are reminiscent of the movement of a person on Free Line skates, and thus the two will share some similar kinematic properties. Specifically, the shape drawn on the floor by the wheels of the quadrapod and a person on Freelines should be the same. This has been qualitatively assessed by comparing Figure 1 to an experiment performed by the team in which a person on freelines skated over waver and then a long sheet of paper – recording the path of the freelines on the paper. Both paths are sinusoidal in nature with a phase offset from the front "foot" to the back "foot".

This project is being constructed as both a Dynamics of Rigid Bodies project and a Design of Microprocessors project. The robot will be propelled by the motion of 8

servo motors which will all be controlled with the combination of a PIC and an FPGA. The servos will bend each joint on the legs of the robot (2 joints per leg) and serve as the driving force behind the robot. The robot built will serve as a physical model used to verify a theoretical model developed by applying concepts from E175 (Dynamics of Rigid Bodies). Setting the angular accelerations and displacements of each limb and recording the forward velocity of the robot with those parameters will serve as a benchmark for the corresponding theoretical model.

The PIC will generate a pattern of pre-determined acceleration and deceleration routines for the servos (4 routines). This pattern will be inputted as an 11-bit signal (3 bits for servo identification and 8 bits for servo position). The FPGA will read and store this signal and output eight separate pulse-modulated signals.

The project has been separated into two sets of goals; a short-term base goal and long-term stretch goal. This serves the purpose to ensure that the Microprocessors component can be completed without the completing the Dynamics of Rigid Bodies portion.

Base Goal

The base goal is to produce a four-legged wheeled robot that can produce translational movement while connected to a benchtop power-source and a computer. Instead of simply connecting the wheels of the robot to motors, the wheels will be free spinning and instead each leg will be controlled by the movement of 2 servos which will control the pitch and yaw of each leg. By attaining this goal we will have proven that we were successful in communicating between the PIC and the FPGA and were able to output the correct output. In order to move the robot across the floor (however ungracefully, slowly or non-linearly) the team should be able to find a satisfactory combination of signals trial and error.

Stretch Goal

Beyond this goal, in conjunction with the Rigid Bodies, the team plans to improve the movement of the robot by precisely determining the angular accelerations and velocities of each servo-motor to generate a more graceful path. Additionally, the team would like to attach a battery pack to the robot and control it via a Bluetooth link to a computer. The user would input text commands (such as <.>,v,^) into HyperTerminal which will then control how the robot moves. This is a stretch goal because the team is concerned about the ability of the servos to carry a battery and also concerned about the amount of power that the servos will be drawing from said battery at once. Additionally, it may be discovered that the servos purchased are simply not strong

enough, and the restricted geometries of the legos may not allow the robot to move in a graceful manner.

NEW HARDWARE

In the Hirose-Fukushima lab in Japan their robot is constructed out of multiple aluminum parts and features a plethora of on-board electronics to control the motion of the robot. While this allows the team in Japan precise control, our team doesn't have the budget or time to construct our robot out of these parts. Instead, our design is constrained to inexpensive-off-the shelf parts.

The physical body of the robot has been assembled using various parts from the Lego Mindstorms NXT 2.0 kit provided by the Harvey Mudd College stockroom. These parts make the majority of the legs, wheels and body of the robot. A summary of the Lego pieces used and how much they weigh can be found in Appendix 1.

To perform the motion of each leg, the team has installed two Futaba 3003 series servos on each leg. The servos are positioned perpendicular to each other and hot-glued onto the Lego legs.

In the future, the team plans to implement a Bluetooth-command protocol, wherein a user can type in a command in HyperTerminal on a computer and the robot will respond. The team plans on using the BlueSMIRF hardware present in lab to connect the breadboard wirelessly to the computer and a simple Bluetooth USB dongle on the computer end.

Lastly, an ambitious goal is to de-tether the robot completely. This de-tethering is unlikely however, because to do so requires the team acquire a battery and breadboard which combined weigh less than 800 grams (the remaining weight the servos can still support) while providing over 2.5 Amps of current to the servos over an appreciable amount of time.

JOINT PROJECT WITH E175: DYNAMICS OF RIGID BODIES

As mentioned before, this project is a joint project with E175: Dynamics of Rigid Bodies. For said class, the team is in the process of performing several calculations and models of the robot to ensure predictable motion of the robot. In properly analyzing all of the mechanics of the device, the team hopes to more precisely and gracefully control the robot by determining exactly which angular accelerations are necessary for constant forward velocity of the machine.

SCHEMATICS

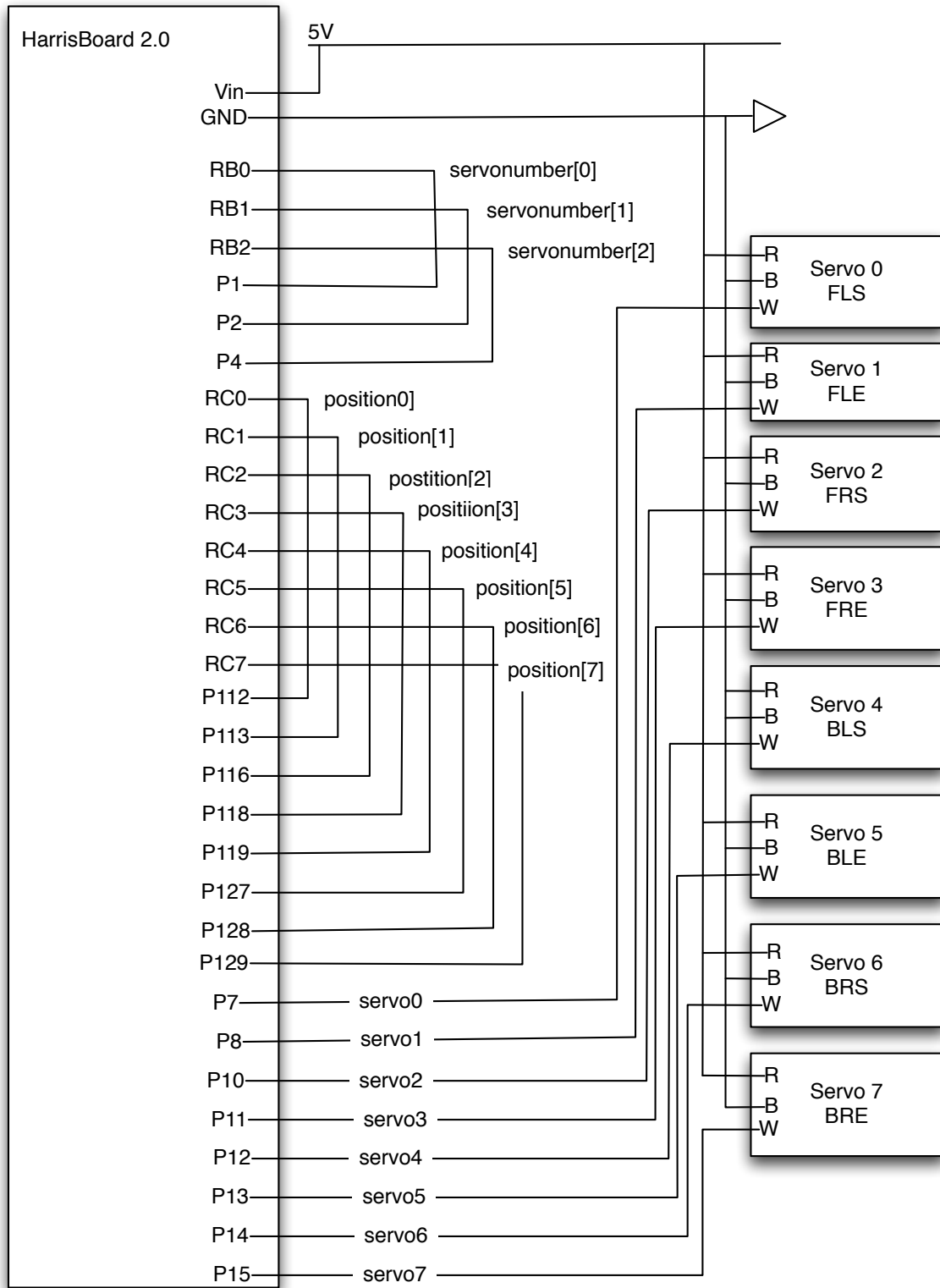


Figure 2 shows the schematic for all the electronics involved in the control unit for the robot.

MICROCONTROLLER DESIGN

We envisioned a design that could be controlled and debugged through C code written for the PIC microcontroller, but could be maintained with the FPGA. From this concept we wrote functions in C to control the elbow and shoulder joint movement as well as the delays necessary to allow the servo to move. These functions were then implemented in different sequences and for different amounts of time in order to create the desired motion of the robot. The main functions used are:

set_servo_position(int servonumber, int position)

This function takes an input of a servo number (0 through 7) and a position (0 to 255) and uses this information to communicate with the FPGA. Because the required pulse widths for the Futaba S3003 servos are values between .5ms and 2.3 ms, the useful position values were discovered to be numbers between 10 and 45. However, this code was designed to create a pulse width of any length and can therefore handle any 8-bit value.

delay_ms(int time)

This function serves the simple purpose of being a counter and halting the system for any amount of time in milliseconds. This was very important because if the servo receives a new position value before it had reached its previous one it will simply move to the most recent position. We did not want this to happen, as it is very important for the motion of our robot that each servo completes its intended motion. Therefore, it is necessary for the PIC to wait a designated amount of time in between servo calls.

FPGA DESIGN

The FPGA serves as the main waveform producing unit on the robot. The FPGA reads 11 bits of information from the PIC (3 bits which identify the servo and 8 bits which identify the position of that servo), and saves this information into the corresponding register. This register is part of a set of 8 registers. Each register corresponds to a different servo, and holds the position that servo should be at.

It then outputs all 8 position registers and creates eight corresponding pulse-modulated waveforms. The output waveforms are all created by an eight-bit counter. The counter increases from 0 to 255 and will remain high from zero until the counter reaches the number stored in the corresponding position register, at which time, the corresponding signal goes low.

Results

Through a combination of close examination of the videos provided by Hirose-Fukushima labs, Rigid Bodies kinematics calculations, and trial and error, the team was able to find the necessary servo positions to create translational motion with the lego

model. As it currently works, the robot moves forward but not in the exact way that the original video indicates. With the Hirose robot, the motion is very smooth and the wheels follow a sine wave on the floor. For the team's recreation, this movement was simplified to a triangular path, which allows the robot to move forward, but in a less graceful manner. Moreover, the team's recreation has a different geometry than the Hirose-Fukushima machine, further constraining the motion it can perform.

None of the stretch goals were met due to time and material constraints. After testing the robot, it was noted that at 5V the robot draws 2 Amps consistently. It was impossible to find a battery that was light and robust enough for the project within a reasonable budget. Moreover, to improve the robot's movement across the floor would have introduced at least 6 more unknowns into the dynamics analysis creating an unsolvable problem, so those efforts were also abandoned. Lastly, the Bluetooth implementation was also forsaken due to time constraints. Some preliminary C-code had been written to perform the predicted functions, but could not be debugged in time for this report.

APPENDIX 1

Bill of Materials. Below are listed all the materials used in the project.

Quantity	Part Name	Part Description
2	Lego Mindstorms NXT 2.0 kit	This kit includes all of lego parts required to built the robot including the tires, hubs, axles, lift arms (straight and bent) and half bushings.
8	Futaba 3300 Servos	These where the 8 servos that controlled the movement of each leg.
5 ft	Red wire	For connecting the Bread Board to the robot.
5 ft	White wire	
5 ft	Black wire	
1	HarrisBoard 2.0	Includes both the PIC and FPGA necessary for controlling the robot and some extra components.
1	Bread Board	For wiring the Harris Board to the servos and ground and power.
2ft	Any colored wire	For wiring between the Bread Board and the Harris Board
1	Power Supply	Must output at least 5 V at 2.5 Amps.

APPENDIX 2

All PIC code used.

```
#include <p18f452.h>
#include <math.h>

#define SERVO0_MIN 30
#define SERVO0_MAX 45
#define SERVO1_MIN 10
#define SERVO1_MAX 45
#define buffer 4

/* function protoypes */
void set_servo_position(int servonumber,int position);
void setup(void);
void delay_us(int time);
void main(void);
void delay_s(int time);
void delay_ms(int time);

// functions
void setup(void){
//T0CON=0b10000001;
T0CON=10000001;
TRISB=0x00;
TRISC=0x00;
TRISD=0x00;
set_servo_position(0,10);
set_servo_position(1,10);
set_servo_position(2,10);
set_servo_position(3,10);
set_servo_position(4,10);
set_servo_position(5,10);
set_servo_position(6,10);
set_servo_position(7,10);
}

void delay_us(int time){ /// time is .8 microseconds
unsigned int t0,tl,th;
TMR0H=0; TMR0L=0;
do{
    tl= TMR0L;
    th=TMR0H;
    t0= tl | th<<8;
    //just wait
}while (t0<time);
}

void delay_s(int time){
int i;
for(i=0;i<(40*time);i++){
    delay_us(100000);
}
}

void delay_ms(int time){
```



```

    int i;
    for(i=0;i<(10*time);i++){
        delay_us(100);
    }
}

void set_servo_position(int servonumber,int position){ // position
between 10 and 45
    PORTB= servonumber;
    PORTC= position;
    //PORTB=buffer;
    //PORTB= buffer;
}

void set_servo_elbow(int servonumber1,int servonumber2,int
servonumber3,int servonumber4, int start_position, int end_position,
int time){
    // 0, 2, 30, 35, 400
    int i;
    int diff;
    int increment;
    diff= end_position-start_position; // 35-30
    increment= time/fabs(diff);
    // PORTD=fabs(diff);

    set_servo_position(servonumber1,start_position); /**
set_servo_position(servonumber2,45-(start_position-10));

    if(diff<0){
        for(i=0;i<fabs(diff);i++){
            set_servo_position(servonumber1,start_position-i);
            set_servo_position(servonumber3,start_position-i);
            set_servo_position(servonumber2,45-(start_position-10)+i);
            set_servo_position(servonumber4,45-(start_position-10)+i);
            delay_ms(increment);
        }
    }
    else{
        for( i=0; i<diff; i++){
            set_servo_position(servonumber1,start_position+i);
            set_servo_position(servonumber3,start_position+i); /***
            set_servo_position(servonumber2,45-(start_position-10)-i);
            set_servo_position(servonumber4,45-(start_position-10)-i);
            delay_ms(increment);
        }
    }
}
// -----
// MAIN!
void main(void){
    setup();
    while(1){
        set_servo_position(1,30);           // shoulder
        set_servo_position(3,25);         // shoulder

        set_servo_position(5,25);
        set_servo_position(7,30);
    }
}

```

```
set_servo_elbow(2,0,6,4,29,34,200); // was 25-30

set_servo_position(1,25);
set_servo_position(3,30);

set_servo_elbow(2,0,6,4,34,29,200);

set_servo_position(1,30); // shoulder
set_servo_position(3,25);

set_servo_position(5,35);
set_servo_position(7,20);

set_servo_elbow(2,0,6,4,29,24,200);

set_servo_position(1,35);
set_servo_position(3,20);

set_servo_elbow(2,0,6,4,29,24,200); // shoulder

}

}
```

APPENDIX 3

All Verilog code used.

```
module PWM(
    input clk,
    input [2:0] servonumber,
    input [7:0] position,
    input reset,
    output reg servo0,
    output reg servo1,
    output reg servo2,
    output reg servo3,
    output reg servo4,
    output reg servo5,
    output reg servo6,
    output reg servo7
);
reg[7:0] counter;
reg[10:0] delay;
reg[7:0] position0;
reg[7:0] position1;
reg[7:0] position2;
reg[7:0] position3;
reg[7:0] position4;
reg[7:0] position5;
reg[7:0] position6;
reg[7:0] position7;

always @ (posedge clk)
begin
    if (servonumber==3'b000) position0=position;
    else if (servonumber==3'b001) position1=position;
    else if (servonumber==3'b010) position2=position;
    else if (servonumber==3'b011) position3=position;
    else if (servonumber==3'b100) position4=position;
    else if (servonumber==3'b101) position5=position;
    else if (servonumber==3'b110) position6=position;
    else if (servonumber==3'b111) position7=position;
    if (counter==position0)servo0=0; // when counter = position,
                                    // servo0 goes low

    if (counter==position1)servo1=0;
    if (counter==position2)servo2=0;
    if (counter==position3) servo3=0;
    if (counter==position4) servo4=0;
    if (counter==position5) servo5=0;
    if (counter==position6) servo6=0;
    if (counter==position7) servo7=0;
    if (counter==255 || reset==1)
        begin
            counter=0;
            servo0=1;
            servo1=1;
            servo2=1;
            servo3=1;
            servo4=1;
        end
end
endmodule
```

```
        servo5=1;
        servo6=1;
        servo7=1;
    end
    if(delay[10]==1)                // check that delay is big enough
                                    // we're sending the servo between
                                    // 50 Hz and 100 Hz
                                    // currently : 76 Hz
        begin
            counter=counter+1;      // increase actual counter
            delay=0;
        end
        delay = delay+1;            // increase delay counter
    end
endmodule
```

WORKS CITED

Farrelly, Ryan. About | Freeline Skates. 2005. 4 October 2009
<<http://www.freelineskates.com/FREELINE/website/aboutus.html>>.

Hirose, Shigeo and Takeuchi Hiroki. "Study on Roller-Walk (Basic Characteristics and its control)."
International Conference on Robotics and Automation. Minneapolis, MN: IEEE, 1996. 3265-3270.

Hirose-Fukushima Robotics Lab. Leg- Wheel Hybrid Walking Vehicle "Roller-Walker". 2006. Tokyo Institute of Technology. 5 October 2009 <http://www-robot.mes.titech.ac.jp/robot/walking/rollerwalker/rollerwalker_e.html>.

kopanda. Latest, an introduction of an advanced snake robot in Japan. 15 March 2006. 4 October 2009
<http://www.youtube.com/watch?v=4cAwzSZqO_w>.